UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/496,844	02/02/2000	Patrick Knebel	10971393-1	6757

22879 7590 12/20/2002

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER
HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 12/20/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/496,844	KNEBEL ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 November 2002* .

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 and 3-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 and 3-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☒ The proposed drawing correction filed on *25 November 2002* is: a)☒ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. Claims 1 and 3-20 have been examined.

Withdrawn Rejections

2. The 35 U.S.C. 103 rejections of claims 1-9 and 12-18 set forth in the previous Office Action, mailed on August 27, 2002, paper number 3, have been overcome by the applicant's amendments and are hereby withdrawn by the examiner.

Maintained Rejections

3. The 35 U.S.C 103 rejections of claims 10 and 11 set forth in the previous Office Action, mailed on August 27, 2002, paper number 3, are respectfully maintained and incorporated by reference.

Response to Arguments

4. Applicant's arguments filed on November 25, 2002, have been fully considered but are not deemed to be persuasive.

5. In the remarks, Applicant argues the rejection of claim 10 on page 6-7, in substance that:

> "...neither Abdallah1 or Abdallah2 teach or suggest dispatching the high-half and low-half operations simultaneously. Both Abdallah references specifically require that the execution of microinstructions occur 'sequentially' in a 'staggered' manner so that the same hardware is used to process both microinstructions."

6. This argument is not found persuasive for the following reasons:

For Applicant's understanding, an example will be discussed. From Fig.4B, at time T, the 32-bit microinstructions ADD X0,Y0 and ADD X1,Y1 are the low half and high half, respectively, of

the SSE instruction ADD X,Y (low 64 bits), where ADD X,Y (low 64 bits) is a sub-instruction of ADD X,Y (which is shown as a 128-bit instruction at time T in Fig.4B). It should be realized from Fig.4A, column 4, lines 39-50, and column 5, lines 23-27, that these microinstructions (ADD X0,Y0 (bits 0-31) and ADD X1,Y1 (bits 32-63)) are dispatched simultaneously. From this passage (column 4, lines 42-45 specifically), it can be seen that the data for ADD X0,Y0 and ADD X1,Y1 is dispatched simultaneously to the two execution units via multiplexers 402 and 404 (or 406 and 408 in case of a "mult"), where they are executed in parallel, as shown in Fig.3 and Fig.4B. In addition, the examiner acknowledges the correctness of Applicant's argument that the Abdallah references use the same hardware to perform staggered execution. However, it should be realized that this is only the case for a 128-bit instruction as shown in Fig.4B. The two 32-bit microinstructions that represent a 64-bit instruction, on the other hand, are dispatched and executed in parallel as discussed above.

7. In the remarks, Applicant argues the rejection of claim 10 on page 6-7, in substance that:

> "...the rejection of claim 10 is improper because neither Abdallah1 nor Abdallah2 disclose or suggest canceling a pending microinstruction in a floating-point unit when 'either' of the microinstructions causes an exception."

8. This argument is not found persuasive for the following reasons:

The use of alternative language such as "either," implies that only one of the alternatives needs to be satisfied. In this situation, the applicant claims "if an exception is taken in either the first or second FP unit, flushing a result in the other FP unit (claim 10(g))." Such a statement would be anticipated by prior art that performs any one of the following:

1) only flushing a result in a second FP unit if an exception is taken in a first FP unit, or

2) only flushing a result in a first FP unit if an exception is taken in a second FP unit, or

3) both flushing a result in a second FP unit if an exception is taken in a first FP unit and flushing a result in a first FP unit if an exception is taken in a second FP unit.

Recall that Abdallah2 has taught the general idea of a system wherein the system is flushed when an exception is generated. See column 10, lines 21-24. Furthermore, in one embodiment of Abdallah2, if an exception were taken in a second microinstruction, the first microinstruction would be flushed. See column 10, lines 24-33. As discussed in the rejection of claim 10 in the previous Office Action, it would have been obvious to cancel each of the microinstructions if an exception were triggered in either one, because it would be pointless to continue executing a subset of an instruction when the overall instruction is improper and any result would be invalid. Also, the applicant argues that the claimed system does not require a "back-off" register or an "undo" mechanism. However, the claim makes no mention of the exclusion of a "back-off" register or "undo" mechanism, and the language of the claim includes the word "comprising," which renders the claim open for the inclusion of unspecified ingredients, even in major amounts. Consequently, this extra component ("back-off" register and/or "undo" mechanism) may exist and still read on the claim.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1, 3-8, 12-17, and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Col et al., U.S. Patent No. 6,330,657 B1 (herein referred to as Col), in view of Hennessy and Patterson, Computer Architecture – A Quantitative Approach, 2nd Edition, 1996 (herein referred to as Hennessy).

11. Referring to claim 1, Col has taught a method for processing software instructions comprising:

a) decomposing a macroinstruction into a plurality of microinstructions. See Fig.4, steps 402 and 404.

b) issuing all of the plurality of microinstructions simultaneously, in parallel. See column 3, lines 52-56.

c) executing all of the plurality of microinstructions simultaneously. See column 3, lines 31-35, and Fig.6 (note in cycle 7 that two microinstructions are executed in parallel).

d) Col has not explicitly taught determining whether an exception occurs in any of the microinstructions, and if an exception occurs in any of the microinstructions, canceling all of the microinstructions. However, exceptions are well known and expected in the art. In general, an exception is an interruption to the normal flow of program control, caused by the program itself or by executing an illegal instruction. Hennessy has shown that some exceptions include I/O device requests, arithmetic underflow and overflow, and memory protection violations. See page 179. Hennessy has further taught the idea of precise exceptions in which the faulting instruction will not change the state of the system. See page 183. In Col's system, since a macroinstruction is broken up into microinstructions, an exception in a single microinstruction would mean that an exception has occurred in the overall macroinstruction, and therefore, all of the microinstructions

that represent a single macroinstruction, should not be able to change the state of the system. Looking at Fig.6 (cycle 7) of Col, a person of ordinary skill in the art would have recognized that the if the BX register held an illegal memory address (ex. a reserved address or an out-of-bounds memory address), then an exception would have to be triggered when the LD T1,[BX] microinstruction is encountered. And, if an exception were triggered, the ADD AX,T1 instruction would have to be cancelled since T1 was not loaded with the appropriate data (i.e. the contents of the memory address stored in BX). If the ADD AX,T1 instruction were allowed to complete, then the instruction would update the system in an incorrect fashion (i.e. the AX register will contain a value that is invalid), which based on the concept taught by Hennessy, is a violation of precise exception implementations. An advantage of this scheme would be to avoid having to undo the changes made by the undesired execution of a microinstruction that is part of a faulty macroinstruction. This will prevent a reduction in throughput in that the extra time required to perform an undo-operation would not be necessary. Therefore, in order to maximize the efficiency of the overall system, it would have been obvious to one of ordinary skill in the art at the time of the invention to determine whether an exception occurs in any of the microinstructions, and if an exception occurs in any of the microinstructions, canceling all of the microinstructions.

12. Referring to claim 3, Col in view of Hennessy has taught a method as described in claim 1. Col has further taught that the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit. From Fig.4, note the integer, floating-point, and SIMD execution units. Multiple execution units are used per clock cycle in order to execute multiple microinstructions. For example, the microinstructions of cycle 7 in

Fig.6 must be executed on different execution units if they are executed in parallel. Then the results of each microinstruction are written to the appropriate storage via store logic (component 420 of Fig.4. See column 17, lines 3-16. Note that the store logic retrieves all of the results from each microinstruction execution and writes the data to the appropriate place. Finally, from Fig.6 (cycle 7), it should be realized that the separate, but parallel, execution of LD T1,[BX] and ADD AX,T1, will produce a result that is expected for the ADD AX,[BX] macroinstruction.

13. Referring to claim 4, Col in view of Hennessy has taught a method as described in claim 1. Col has further taught that all of the microinstructions are executed on the same clock cycle. See Fig.6, cycle 7, for instance.

14. Referring to claim 5, Col in view of Hennessy has taught a method as described in claim 1. Col has further taught that his system can execute floating-point operations, indicative of the floating-point. Col has not explicitly taught that the microinstructions are executed over multiple clock cycles. However, it is well known and expected in the art that floating point operations can consume more than one clock cycle for execution purposes. Hennessy has disclosed this concept on page 187. Hennessy has also shown a pipeline that accommodates floating-point execution through multiple execution stages. See Fig.3.44 and Fig.3.45 on page 190. Since it has been disclosed by Hennessy that a floating-point operation takes multiple instruction execution cycles, it follows that it would have been obvious to one of ordinary skill in the art at the time of the invention to use a pipelined floating-point execution unit with multiple execution stages if floating-point operations are desired.

15. Referring to claim 6, Col in view of Hennessy has taught a method as described in claim 1. Col has not explicitly taught the implementation of this method in a system emulating SSE

instructions. However, a person of ordinary skill in the art would have realized that SIMD is often associated with accelerating floating-point performance in general purpose processors mainly due to the fact that multiple floating-point operations can be executed in parallel. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to incorporate SSE instructions into Col's system in order to maximize the efficiency in which the processor executes floating-point operations.

16. Referring to claim 7, Col in view of Hennessy has taught a method as described in claim 6. Col has not explicitly taught that the system allows a single instruction to operate on multiple single-precision floating-point values. However, Hennessy has shown that single-precision floating-point numbers are well known and expected in the art. In fact, the 32-bit single-precision floating-point number format is an IEEE standard. See page A-15 to A-16. Therefore, because it is so common and well known in the art, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement such a format within Col's system.

17. Referring to claim 8, Col in view of Hennessy has taught a method as described in claim 1. Col has not explicitly taught that a flag is updated based upon a result of the execution of the microinstructions. However, it is well known in the art that processors contain a status register. The status register contains bits that are set or cleared based on the result of an operation. Some of the more common flags are ones that indicate a result of zero, a negative number, and overflow. These flags can then be checked in conditional situations, such as branches. Therefore, it would have been obvious to one of ordinary skill in the art to update a flag based upon the result of the execution of the microinstructions.

18. Referring to claim 12, Col has taught a computer system comprising:

a) a processor comprising:

1) a floating-point unit. See Fig.4, component 414.

2) Col has not explicitly taught a computer system with a ROM. However, Official Notice is taken that ROMs are well known and expected in the art. Processors contain Read-Only Memory to store essential software of the computer. Because it's non-volatile memory, ROM does not lose its contents when the power is turned off. Therefore, a ROM chip is used to store control programs for the computer, such as the bootstrap program (which tells the computer how to start and load the operating system) and other types of configuration information.

3) a plurality of floating-point registers. See column 5, lines 49-52.

b) wherein the processor is configured to emulate an instruction set by:

1) decomposing a macroinstruction into a plurality of microinstructions. See Fig.4, steps 402 and 404.

2) issuing all of the plurality of microinstructions simultaneously, in parallel. See column 3, lines 52-56.

3) Col has not explicitly taught determining whether an exception occurs in any of the microinstructions, and if an exception occurs in any of the microinstructions, canceling all of the microinstructions. However, recall from the rejection of claim 1(d) above, that it would have been an obvious modification to add exception detection functionality to Col's system (in view of Hennessy). This modification, as described above, would allow for the detection of an exception within a single microinstruction and if an exception is

triggered, the other parallel instructions will be subject to cancellation. Therefore, this portion of claim 12 is rejected for the reasons set forth in the rejection of claim 1(d) above.

19. Referring to claim 13, Col in view of Hennessy has taught a computer system as described in claim 12. Col has further taught that the processor is further configured to emulate the instruction set by executing all of the microinstructions. See column 3, lines 31-35, and Fig.6 (note in cycle 7 that two microinstructions are executed in parallel).

20. Referring to claim 14, Col in view of Hennessy has taught a computer system as described in claim 13. Furthermore, claim 14 is rejected for the same reasons set forth in the rejection of claim 3 above.

21. Referring to claim 15, Col in view of Hennessy has taught a computer system as described in claim 14. Furthermore, claim 15 is rejected for the same reasons set forth in the rejection of claim 8 above.

22. Referring to claim 16, Col in view of Hennessy has taught a computer system as described in claim 15. Col has not explicitly taught determining whether an exception occurs in the execution of any of the microinstructions, and if an exception occurs, causing the exception to cancel all of the microinstructions. However, recall from the rejection of claim 1(d) above, that it would have been an obvious modification to add exception detection functionality to Col's system (in view of Hennessy). This modification, as described above, would allow for the detection of an exception within a single microinstruction and if an exception is triggered, the other parallel instructions will be subject to cancellation. Furthermore, Hennessy has disclosed that the stopping of instructions before a system update is performed is based on an exception

being detected during instruction execution. See page 182. Therefore, this portion of claim 12 is rejected for the reasons set forth in the rejection of claim 1(d) above, and for the teachings of Hennessy on page 182.

23. Referring to claim 17, Col in view of Hennessy has taught a computer system as described in claim 12. Furthermore, claim 17 is rejected for the same reasons set forth in the rejection of claim 6 above.

24. Referring to claim 19, Col in view of Hennessy has taught a method as described in claim 1.

a) Col has further taught that the step of issuing comprises forcing the microinstructions to issue simultaneously, in lockstep with each other. See column 3, lines 52-56.

b) Col has not explicitly taught that the step of canceling comprises canceling all of the plurality of microinstructions without regard to the relative ages of the microinstructions and without using a backoff mechanism. However, recall from the rejection of claim 1(d) above, that it would have been an obvious modification to add exception detection functionality to Col's system (in view of Hennessy). This modification, as described above, would allow for the detection of an exception within a single microinstruction and if an exception is triggered, the other parallel instructions will be subject to cancellation. Furthermore, from the rejection of claim 1(d) above, Hennessy has disclosed the implementation of precise exceptions in which the faulty instruction is prevented from finishing. Therefore, the microinstructions that have been spawned from a single macroinstruction would be unable to modify the state of the system (i.e. write invalid results to registers or memory locations). Since the system is not updated by these instructions, there is no need for a backoff mechanism to undo any changes that have been made.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to cancel instructions without the use of a backoff mechanism.

25. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Col in view of Hennessy and further in view of Phillips et al., U.S. Patent No. 6,038,652 (as applied in the previous Office Action and herein referred to as Philips).

26. Referring to claim 9, Col in view of Hennessy has taught a method as described in claim 1.

a) Col has not explicitly taught that if an unmasked exception occurs, canceling the execution of all of the plurality of microinstructions, without regard to the relative ages of each of the plurality of microinstructions, and invoking a microcode handler. However, recall from the rejection of claim 1(d) above, that it would have been an obvious modification to add exception detection functionality to Col's system (in view of Hennessy). This modification, as described above, would allow for the detection of an exception within a single microinstruction and if an exception is triggered, the other parallel instructions will be subject to cancellation. In addition, it is well known and expected in the art that the triggering of an exception will result in the invocation of a microcode handler. See Hennessy, page 181, last paragraph. In general, the handler is invoked in order to correct the cause and effects of the exception and allow the processor to continue execution. Therefore, in order to correctly service an exception, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a microcode handler, which must be invoked upon exception detection.

b) Col in view of Hennessy has not explicitly taught updating at least one exception flag (when an unmasked exception occurs) by independently generating a logical OR of exceptions for a plurality of functional units. However, Phillips has taught the concept of simultaneously checking SIMD elements for exceptions and combining each individual exception into an overall exception. See FIG.2. Furthermore, the combining element (230) in FIG.2 can be implemented as an OR gate that generates a flag (240) used to specify whether or not an exception has occurred. See column 3, lines 60-63. A person of ordinary skill in the art would have recognized that the concept of Phillips would be applicable in Col's system in order to check for exceptions during the parallel execution of microinstructions. In a SIMD processor (as taught by Col), the overhead incurred to process the many possible exceptions generated by SIMD elements may be expensive and lead to degradation in performance. The system of Philips provides an efficient technique to report exceptions occurring in computing complex functions on a SIMD machine. An advantage to this scheme is that since an exception flag is produced according to a parallel execution of microinstructions as opposed to a serial execution of microinstructions, the processor will be able to detect an exception sooner and therefore sooner make the determination that the involved microinstructions should be cancelled. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to update at least one exception flag in Col's system based on the exception check for a plurality of microinstructions.

27. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Col in view of Hennessy, as applied to claims 12 and 17 above, and further in view of Makineni et al., U.S.

Patent No 6,321,327 B1 (as applied in the previous Office Action and herein referred to as Makineni).

28. Referring to claim 18, Col in view of Hennessy has taught a computer system as described in claim 17. Col has further taught that the SIMD execution units perform operations (such as an add) on multiple operands from a first SIMD register with corresponding multiple operands from a second SIMD register. See column 12, lines 1-8. Col has not explicitly taught the use of two 82-bit FP registers for emulating four 32-bit single-precision floating-point values in an SSE register. However, Makineni has taught the use of 82-bit registers to hold 32-bit single precision floating-point numbers. See Fig.2B and Fig.3. Since SIMD operations involve performing a single operation on multiple pairs of data elements, it follows that multiple pairs of operands must be available to the SIMD execution unit. A person of ordinary skill in the art would have recognized that by implementing 82-bit registers, multiple pairs of operands would be supplied to a SIMD execution unit by using multiple registers. In addition, by packing more than one data element into a single register, the amount of addressable registers could be decreased, resulting in less wires used for addressing purposes. Finally, each of the standard IEEE floating-point formats can be specified through the use of a single 82-bit FP register, allowing the system to operate on different-precision operands depending on the situation. See the floating-point standards on page A-13 of Hennessy and note Fig.2A of Makineni shows a double-extended precision floating-point number. Therefore, in order to decrease the amount of hardware, while assuring the system has the capability of processing a wide variety of floating-point numbers, it would have been obvious to one of ordinary skill in the art to use two 82-bit FP registers for emulating four 32-bit single-precision floating-point values in an SSE register.

29. Claim 20 is rejected under 35 U.S.C. 103(a) as being unpatentable over Abdallah1 (U.S. Patent No. 6,233,671 B1) in view of Abdallah2 (U.S. Patent No. 6,085,312), as applied to claim 10 in the previous Office Action, mailed on August 27, 2002, paper number 3.

30. Referring to claim 20, Abdallah1 in view of Abdallah2 has taught a method as described in claim 10.

a) Abdallah1 has further taught that the step of forcing the high-half and low-half operations to issue in parallel comprises causing the high-half and low-half operations to execute simultaneously in lockstep with each other. From Fig.4B and the discussion above regarding the rejection of claim 10, it can be seen that the two microinstructions ADD X0,Y0 and ADD X1,Y1 begin to execute in parallel at time T and finish at time T+1.

b) Abdallah1 has not explicitly taught that the step of flushing a result comprises canceling each of the high-half and low-half operations if an exception is taken in either the first or second FP unit. However, from the discussion above regarding the rejection of claim 10, Abdallah2 has taught a system in which if an exception is generated by a second microinstruction, then a first microinstruction will be flushed from the system. See column 10, lines 24-32. Abdallah2 has taught the general idea of flushing a remaining microinstruction when another microinstruction from the same group (i.e. from the same macroinstruction) generates an exception. A person of ordinary skill in the art would have recognized that if an exception (such as divide-by-zero, overflow, underflow, etc.) were triggered by one microinstruction, then continuing to process another microinstruction would be pointless. The result that this non-exception-generating microinstruction would produce would be irrelevant since there is already an error in the overall macroinstruction (since one of its microinstructions erred). Therefore, it would have been

obvious to one of ordinary skill in the art at the time of the invention to cancel each of the high-half and low-half operations if an exception is taken in either the first or second FP unit.

Conclusion

31. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

DJH
David J. Huisman
December 17, 2002

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100